Lecture 1: Introduction

Welcome to Berkeley Computer Science!

Humans of CS 61A

2 Lecturers
12 TAs
13 Tutors

100+ Lab assistants!
400+ Students!!!

Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems
   Artificial Intelligence
   Security
   Networking
   Theory
   Computational Biology
   ...

Natural Language Processing
   Machine Learning
   Computer Vision
   Planning
   Robotics
   Perception
   Manipulation
   Navigation and Locomotion
   Human-Robot Interaction
   ...

CS 61A in one slide

- High-level ideas in computer science:
  - Abstraction: manage complexity by hiding the details
  - Paradigms: utilize different approaches to programming

- Master these ideas through implementation:
  - Learn the Python programming language (& others)
  - Complete large programming assignments

- A challenging course that will demand a lot from you

Alternatives to CS 61A

CS 10: The Beauty and Joy of Computing
cs10.org  Offered this summer!

Data Science 8: Foundations of Data Science
data8.org
Course Policies
Details on cs61a.org

Course overview

- Lectures: Mon-Thurs, 11am-12:30pm, 2050 VLSB
- Labs: the most important part of this course
- Discussions: the most important part of this course
- Office hours: the most important part of this course
- Online textbook: composingprograms.com

- Regular homework assignments
- 4 big programming projects
- Weekly quizzes, one midterm, and one final exam
- Lots of special events!

Grading

<table>
<thead>
<tr>
<th>Component</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>30</td>
</tr>
<tr>
<td>Weekly quizzes</td>
<td>40</td>
</tr>
<tr>
<td>Lab</td>
<td>20</td>
</tr>
<tr>
<td>Midterm</td>
<td>40</td>
</tr>
<tr>
<td>Projects</td>
<td>100</td>
</tr>
<tr>
<td>Final</td>
<td>70</td>
</tr>
</tbody>
</table>

A few grading details

- 10 homework assignments, 3 points each
  - Can make up points from one homework with surveys
- 12 (graded) lab assignments, 2 points each
  - Two lowest lab scores will be dropped
- Written quizzes will be in lecture on Thursdays
  - We have sent out instructions for students who cannot attend Thursday lectures
  - One written or coding quiz score will be dropped
- This class is not curved!
  - Collaboration, not competition

The limits of collaboration

- Everyone should give and receive help, because everyone benefits and learns
- There is only one rule:
  - Your code is yours, and yours only.
- This means that:
  - You cannot copy or use code from anyone except your partner
  - You cannot share your code with anyone except your partner
- Share and discuss ideas, not code
- Build good habits now!

Getting help

- Discuss everything in the course, except exams, with your partner and your classmates
  - Teaching is the best way to learn
- Ask and answer questions on Piazza
- Use the course staff! We’re here to help you learn
  - Labs and office hours are the perfect time to talk to the lecturers, TAs, tutors, and lab assistants
  - Lab assistants will also be available for checkoffs during labs
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
  • Grading
    • 75% of students in this course receive As and Bs
    • There is no curve! All of you can get an A+
  • Cheating
    • There is a community of staff and students that want you to succeed, and will help you succeed

• The most important course policy is learning
• Learn a lot, have fun, and welcome to 61A!

An Introduction to Programming

And, conveniently, an introduction to Python

Course organization

• Every week will center around a theme, and have a specific set of goals.

  Introduction → Functions → Data → Mutability

  Objects → Interpretation → Paradigms → Applications

• This week (Introduction), the goals are:
  • To learn the fundamentals of programming
  • To become comfortable with Python

What’s in a program? (demo)

• Programs work by manipulating values

• Expressions in programs evaluate to values
  • Primitive expressions evaluate directly to values with minimal work needed

• Operators combine primitives expressions into more complex expressions

• The Python interpreter evaluates expressions and displays their values

Mathematical expressions (demo)

\[
\lim_{x \to \infty} \frac{1}{x} \quad sgn(x) \quad \sin x \\
\sum_{i=1}^{n} \sqrt{x} \quad x^y \quad \ln x \\
\binom{x}{y} \quad \frac{x}{y} \quad |x| \quad x + y \\
x \mod y
\]

Call expressions

add (2, 3) operands

• In a call expression, the operator and operands themselves are expressions

• To evaluate this call expression:
  1. Evaluate the operator to get a function
  2. Evaluate the operands to get its values
  3. Apply the function to the values of the operands to get the final value
Nested call expressions

\[
\text{add}(\text{add}(2, \text{mul}(4, 6)), \text{mul}(3, 5))
\]

- What does this call expression evaluate to?
- What are the steps that the Python interpreter goes through to evaluate this expression?